#

# Copyright (c) 2008 Texas Instruments and others.

# All rights reserved. This program and the accompanying materials

# are made available under the terms of the Eclipse Public License v1.0

# which accompanies this distribution, and is available at

# http://www.eclipse.org/legal/epl-v10.html

#

# Contributors:

# Texas Instruments - initial implementation

#

# \*/

#

# ======== xdc\_top.mak ========

# Top-level make file to (re)build multiple XDC packages.

#

# To (re)build all packages from scratch type the following:

# gmake -f xdc\_top.mak PACKAGES="..." clean

# gmake -f xdc\_top.mak PACKAGES="..."

#

# To run tests type the following:

# gmake -f xdc\_top.mak PACKAGES="..." test

#

# To (re)build all libraries (not executables or dlls) type the following:

# gmake -f xdc\_top.mak PACKAGES="..." .libraries

#

#

# ======== HOSTOS ========

# If not specified in environment or on the command line, try to determine

# the host OS

#

export HOSTOS ?= $(if $(findstring :,$(WINDIR)$(windir)$(COMSPEC)$(comspec)),Windows,Solaris)

#

# Windows shell is useless; we need a more modern shell that has string

# substitution capabilities; e.g., dir=${file%/\*}. In this case, we use

# cygwin's ash.exe.

#

ifeq (Windows,$(HOSTOS))

 SHELL := $(XDCROOT)/bin/sh.exe

endif

#

# ======== command definitions ========

# The following commands are used in this make file. We define

# macros for these commands because we want the makefiles to be

# portable to Korn shell's AND Win2k DOS shells; Win2k shells have

# "date" as a built-in command, so it is not possible to find

# date.exe along the path!

#

# DATE = get the current date and time

# ECHO = echo arguments

#

ifeq (Windows,$(HOSTOS))

 STDERR := $(subst \,/,"$(XDCROOT)"/bin/stderr)

 ECHO = $(STDERR) echo

 DATE = $(if $(TFLAG),$(STDERR) echo,$(STDERR) date)

else

 ECHO = echo

 DATE = $(if $(TFLAG),echo,date)

endif

#

# We make the following overridable because the names of these

# utilities may vary (xs.x86U verses xs, for example)

#

tools := $(firstword $(wildcard $(XDCROOT)/TOOLS $(XDCROOT)/etc/TOOLS))

ifneq (,$(tools))

 include $(tools)

endif

FINDPKGS ?= $(XDCROOT)/bin/xdcpkg

ifeq (Windows,$(HOSTOS))

 ifeq (sh.exe,$(SHELL))

 FINDPKGS := $(subst /,\,$(FINDPKGS))

 endif

endif

FINDPKGPATH := $(if $(XDCPATH),$(XDCPATH);)$(XDCROOT)/packages;^

empty :=

space := $(empty) $(empty)

# convert to absolute path so sub-makes see a consistent config.bld

\_TMPCFG := $(XDCBUILDCFG)

override \_XDCBUILDCFG =

ifdef XDCBUILDCFG

 # either pass XDCBUILDCFG on the sub-make command line or in the environment

 ifeq (environment,$(origin XDCBUILDCFG))

 export XDCBUILDCFG

 else

 override \_XDCBUILDCFG = XDCBUILDCFG="$(XDCBUILDCFG)"

 endif

 # There is a bug in make 3.81, where abspath returns an incorrect path

 # when given an absolute path containing ":". Once the bug is fixed,

 # only the code in 'else' should be retained.

 ifeq (Windows,$(HOSTOS))

 # This path2dos invocation is here in case XDCBUILDCFG already has

 # spaces. Spaces would break the logic that separates the volume

 # and the path on the volume. The case where $abspath introduces spaces

 # is handled in xdc\_rules.mak.

 override XDCBUILDCFG := $(shell "$(XDCROOT)/packages/xdc/services/io/release/path2dos.exe" "$(XDCBUILDCFG)")

 \_SEP := $(subst :, ,$(subst \,/,$(XDCBUILDCFG)))

 \_NODRIVE := $(lastword $(\_SEP))

 \_DRIVE := $(firstword $(\_SEP))

 override XDCBUILDCFG := $(abspath $(\_NODRIVE))

 # Check if we actually had ":" in the original XDCBUILDCFG, and if

 # abspath returned a path without ":" (C:/absolute/volume/path).

 ifneq ($(\_DRIVE),$(\_NODRIVE))

 ifeq ($(findstring :,$(XDCBUILDCFG)),)

 override XDCBUILDCFG := $(\_DRIVE):$(XDCBUILDCFG)

 endif

 endif

 else

 override XDCBUILDCFG := $(abspath $(XDCBUILDCFG))

 endif

endif

# in case $(abspath ...) fails or is undefined, restore XDCBUILDCFG

ifndef XDCBUILDCFG

 ifdef \_TMPCFG

 override XDCBUILDCFG := $(\_TMPCFG)

 endif

endif

#

# ======== macro definitions ========

# GOALS = list of all "known" make goals

#

# ALLGOALS = list of all known and command line specified make goals

#

# KFLAG = k if the -k option to make was set, otherwise "". This is

# used by subrule (below) to determine whether to continue

# building other packages when an error occurs.

#

# QFLAG = q if the -q option was set in XDCOPTIONS, otherwise "".

# This is used below to determine whether to display banners

# before and after recursive builds.

#

# TFLAG = t if the -t option was set in XDCOPTIONS, otherwise "".

# This is used below to determine whether to display dates

# in banners; if t is specified, then do \*not\* display dates.

#

# subdirs = list of all sub-directories containing a build script

#

# comma = the comma character; so we can use comma in other commands

# without confusing make

#

# subgoals = list of phony goals for each directory in subdirs with the

# specified suffix

#

# subrule = rule for making a subgoal: recursive invocation of make in

# the specified subdirectory to make the specified goal

#

# allrules = list of rules and dependencies for each goal in ALLGOALS

#

GOALS = .interfaces .libraries .dlls .executables test all release clean

ALLGOALS = $(sort $(GOALS) $(MAKECMDGOALS))

KFLAG := $(findstring k,$(firstword $(filter-out --%,$(MAKEFLAGS))))

QFLAG := $(findstring q,$(XDCOPTIONS))

TFLAG := $(findstring t,$(XDCOPTIONS))

#

# ======== comma ========

#

comma=,

#

# ======== unique ========

# Remove duplicate words from a list and preserve the order of the words

# in the list

#

unique = $(if $(strip $(1)),$(firstword $(1)) $(filter-out $(firstword $(1)),$(call unique, $(wordlist 2,$(words $(1)),$(1)))))

#

# ======== noreflex ========

# Filter out reflexive pairs (for example, "a;a")

#

noreflex = $(foreach pair,$(1),$(if $(filter-out $(firstword $(subst ;, ,$(pair))), $(subst ;, ,$(pair))),$(pair)))

#

# ======== \*PACKAGES ========

# Ensure they are defined on the command line or in a makefile and \*not\*

# in the environment; this allows recursive invocations of "xdc -P\*" with

# different -P\* options.

#

ifneq (,$(findstring environment,$(origin PACKAGES)))

 override PACKAGES :=

endif

ifneq (,$(findstring environment,$(origin RPACKAGES)))

 override RPACKAGES :=

endif

ifneq (,$(findstring environment,$(origin rPACKAGES)))

 override rPACKAGES :=

endif

ifneq (,$(findstring environment,$(origin DPACKAGES)))

 override DPACKAGES :=

endif

#

# ======== spaces ========

# On Windows, convert spaces in pathnames into 8.3 DOS short pathnames.

#

ifneq ($(findstring $(space),$(PACKAGES)),)

 ifeq (Windows,$(HOSTOS))

 override PACKAGES := $(shell "$(XDCROOT)/packages/xdc/services/io/release/path2dos.exe" "$(PACKAGES)")

 endif

endif

ifneq ($(findstring $(space),$(RPACKAGES)),)

 ifeq (Windows,$(HOSTOS))

 override RPACKAGES := $(shell "$(XDCROOT)/packages/xdc/services/io/release/path2dos.exe" "$(RPACKAGES)")

 endif

endif

ifneq ($(findstring $(space),$(rPACKAGES)),)

 ifeq (Windows,$(HOSTOS))

 override rPACKAGES := $(shell "$(XDCROOT)/packages/xdc/services/io/release/path2dos.exe" "$(rPACKAGES)")

 endif

endif

ifneq ($(findstring $(space),$(DPACKAGES)),)

 ifeq (Windows,$(HOSTOS))

 override DPACKAGES := $(shell "$(XDCROOT)/packages/xdc/services/io/release/path2dos.exe" "$(DPACKAGES)")

 endif

endif

#

# ======== semicolons ========

# Up to this point, the packages on the command line have been delimited by

# semicolons. Change to spaces to support gmake string functions.

#

override PACKAGES := $(subst ;, ,$(PACKAGES))

override RPACKAGES := $(subst ;, ,$(RPACKAGES))

override rPACKAGES := $(subst ;, ,$(rPACKAGES))

override DPACKAGES := $(subst ;, ,$(DPACKAGES))

#

# ======== subdirs ========

# list of all sub-directories containing a build script

#

ifneq (,$(strip $(PACKAGES)))

 subdirs := $(sort $(dir $(wildcard $(subst //,/,$(addsuffix /package.bld,$(strip $(PACKAGES)))))) )

else

 ifneq (,$(strip $(RPACKAGES)))

 subdirs := $(sort $(shell "$(FINDPKGS)" $(RPACKAGES)))

 else

 ifneq (,$(strip $(DPACKAGES)))

 subdirdeps := $(shell "$(FINDPKGS)" -p "$(FINDPKGPATH)" -d -m\; $(DPACKAGES))

 subdirs := $(sort $(subst ;, ,$(subdirdeps)))

 subdirdeps := $(call noreflex,$(subdirdeps))

 else

 ifneq (,$(strip $(rPACKAGES)))

 subdirs := $(sort $(shell "$(FINDPKGS)" -r $(rPACKAGES)))

 endif

 endif

 endif

endif

ifeq (,$(subdirs))

define NOPKGS

Warning: nothing to build; the directories named after -P[RrD] don't contain any buildable packages

 -P $(PACKAGES)

 -Pr $(rPACKAGES)

 -PR $(RPACKAGES)

 -PD $(DPACKAGES)

endef

 $(warning $(NOPKGS))

endif

#

# ======== subgoals ========

# list of all phony "sub goals" necessary to build the specified goal in

# each valid sub-directory (subdirs).

#

subgoals = $(foreach d,$(subdirs),$(d)$(comma)$(1))

#

# ======== subrule ========

# Rule template for each sub-goal

#

# Params:

# $(1) goal

# $(2) sub-directory

#

define subrule

.PHONY: $(2),$(1)

$(1): $(2),$(1)

$(2),$(1):

 @$(if $(QFLAG),,echo ======== $(1) [$(2)] ========)

 +$(if $(KFLAG),-)@"$(MAKE)" -r -R -C $(2) --no-print-directory -f "$(XDCROOT)/packages/xdc/bld/xdc.mak" XDCROOT="$(XDCROOT)" HOSTOS=$(HOSTOS) $(\_XDCBUILDCFG) $(1)

endef

#

# ======== allrules ========

#

allrules = $(foreach G,$(sort $(ALLGOALS)),$(foreach d,$(subdirs),$(call subrule,$(G),$(d))))

#

# Check that the version of make is strictly greater than 3.79.99

#

ifeq (3.79.99,$(word 2, $(sort 3.79.99 $(MAKE\_VERSION))))

 $(error this makefile requires GNU make version 3.80 or greater.)

endif

#

# ======== tsufs ========

# The list of all target specific qualifiers referenced on the command line.

#

# For example, tsufs = "64P 86U" for MAKECMDGOALS = "all,86U all,64P test,86U"

#

tsufs := $(sort $(foreach g,$(GOALS),$(subst $(g)$(comma),,$(filter $(g)$(comma)%,$(MAKECMDGOALS)))))

#

# ======== tdeps ========

# Template for dependencies among specified target qualified goals

#

# Params

# $(1) target qualifier suffix

#

define tdeps

all,$(1) $(call subgoals,all$(comma)$(1)): .executables,$(1)

test,$(1) $(call subgoals,test$(comma)$(1)): .executables,$(1)

release,$(1) $(call subgoals,release$(comma)$(1)): .executables,$(1)

.dlls,$(1) $(call subgoals,.dlls$(comma)$(1)): .libraries,$(1)

.libraries,$(1) $(call subgoals,.libraries$(comma)$(1)): .interfaces

.executables,$(1) $(call subgoals,.executables$(comma)$(1)): .libraries,$(1)

.executables,$(1) $(call subgoals,.executables$(comma)$(1)): .dlls,$(1)

ALLGOALS += $(foreach g,$(GOALS),$(g),$(1))

endef

#

# ======== alldirdeps ========

#

abstgoals=$(sort $(ALLGOALS) $(foreach t,$(tsufs),$(foreach g,$(GOALS),$(g),$(t))))

dirdeps=$(foreach G,$(abstgoals),$(call dirdep,$(1),$(2),$(G)))

define dirdep

$(1),$(3) : $(2),$(3)

endef

alldirdeps = $(foreach p,$(subdirdeps),$(call dirdeps,$(word 1,$(subst ;, ,$(p))),$(word 2,$(subst ;, ,$(p)))))

#

# ======== alltdeps ========

#

alltdeps = $(foreach t,$(tsufs),$(call tdeps,$(t)))

#

# ======== goal dependencies ========

#

all $(call subgoals,all): .executables

test $(call subgoals,test): .executables

release $(call subgoals,release): .executables

.executables $(call subgoals,.executables): .libraries .dlls

.dlls $(call subgoals,.dlls): .libraries

.libraries $(call subgoals,.libraries): .interfaces

$(eval $(alltdeps))

$(eval $(alldirdeps))

.PHONY: $(ALLGOALS)

ifeq (,$(QFLAG))

.PHONY: \_startime

\_startime:

 @$(ECHO) making $(if $(MAKECMDGOALS),$(MAKECMDGOALS),all): `$(DATE)` ...

$(ALLGOALS) $(foreach G,$(ALLGOALS),$(call subgoals,$(G))) : \_startime

$(filter-out clean,$(sort $(ALLGOALS))):

 @$(ECHO) $@ files complete: `$(DATE)`.

clean:

 @$(ECHO) cleaning complete: `$(DATE)`.

endif

#

# ======== goal rules ========

#

$(eval $(allrules))

#

# @(#) xdc.bld; 1, 0, 2,475; 3-20-2014 17:05:11; /db/ztree/library/trees/xdc/xdc-A32x/src/packages/

#